A Fully Abstract Model for Mobile Ambients

M. Coppo M. Dezani-Ciancaglini

Dipartimento di Informatica, Università degli Studi di Torino
corso Svizzera 185, 10149 Torino, Italia,
{coppo,dezani}@di.unito.it

Abstract

Aim of this paper is to investigate the possibility of developing filter models for calculi representing mobility. We will define a model for a variant of the Ambient Calculus. This model turns out to be fully abstract with respect to a notion of contextual equivalence which takes into account the ambients at top level.

1 Introduction

The *Ambient Calculus* [7], a calculus of computation that allows active processes to move between sites and interact with then, has been successfully proposed as a model for the Web. Owing to its interest a number of studies on various foundational aspects of this or derived systems have been recently developed. The subjects of these investigations have been mainly type systems (finalized to the proof of various properties like safe communications [8] or security [6,14,5]), proof systems [9], abstract interpretations [16] and flow analysis [11]. No attempt has been done, up to now, of defining denotational-like models of the calculus.

One main difficulty in defining models of the Ambient Calculus is that of finding an abstract counterpart to the notion of mobility. A promising tool for overcoming this difficulty seems the notion of "logical" semantics in which domains are described by abstract filters of logical formulas expressing properties of the terms of the calculus. The calculus itself can then provide the basic tools for the description of its semantics.

In this paper we investigate the possibility of developing filter models for calculi representing mobility. We will define, in particular, a model for a variant of the Ambient Calculus. This model turns out to be fully abstract with respect to the notion of contextual equivalence defined in [15].

The "logical" approach to denotational semantics goes back to [21], and has been advocated in [1] as a general paradigm unifying, among other things, type as-

[1] Partially supported by MURST Cofin '99 TOSCA.

signments, logic of programs and logical characterization of processes behaviors, such as Hennessy-Milner logic. This semantics can be introduced through a type assignment system, with types ordered by an inclusion relation. Types are intended as logical formulas representing properties of terms, and type inclusion as implication. The interpretation of a term corresponds then to the filter of all types that can be assigned to it. The set of all filters of types (which is a domain) determines a denotational model of the calculus, in the sense that the denotation of a term can be given in a compositional way. This construction has been first applied in [3] and [2] to obtain models of the λ-calculus. In the same line are the studies concerning extensions of λ-calculus by means of operators with concurrent features like [20,4,12,13]. In [4], in particular, the intersection type operator is seen as the basic tool to represent nondeterministic choice in the "may" perspective.

In [18] Hennessy presents the first denotational model of higher-order concurrent processes based on a compromise between type systems and modal logic. The resulting filter model turns out to be fully abstract with respect to an operational semantics based on a notion of testing and "may" convergence. A similar result has been proved in [17] for a kernel of the language FACILE. A filter model for higher-order processes which is adequate but not complete with respect to the "must" testing as been proposed in [19]. The same approach is used in [10] to build a filter model of the π-calculus which is fully abstract for "may" convergence.

In this paper we consider a language called the *Selfopening* Ambient Calculus, which extends the system of [7] by increasing the opening capabilities of processes. While in the Ambient Calculus a process can open only processes running at its same level, in our system a process has also the capability, exercising a new action called so, of opening its enclosing ambient, rising itself one level up. The so action is strongly related to the *acid* operation of [7] and it is not internally representable in the standard Ambient Calculus. However, as it will be remarked in the conclusion, our model is also adequate for the standard Ambient Calculus, but in this case completeness fails.

The type system used for the definition of the model can also be seen as a proof system to express ambient and process properties. Proof systems with these aims have also proposed by Cardelli and Gordon (see e.g. [9]). In particular [9] introduces a proof system including modal operators. The logic language of [9] is quite different from ours since it is more powerful in expressing properties of ambients, but it fails to directly express properties like "P is able to perform an $open$ action". Moreover the system of [9] can express intensional properties (like P *is* the null process) and it is then not suitable to be taken as a basis for the construction of a model of contextual semantics, where properties need to have an extensional meaning.

2 The Language

The calculus of Selfopening Mobile Ambients is an extension of the calculus of Mobile Ambients [7]. In the standard Ambient Calculus there is a symmetry in the

moving capabilities (an ambient can entering and exiting other ambients), but the opening action has no symmetrical counterpart. A process P can open an ambient b occurring at its same level (i.e. concurrent with b) by exercising an action $open\, b$. But a process has no way of opening the ambient a in which he is running raising itself one level up. This kind of action, allowed by the so primitive, seems indeed reasonable in a distributed environment, in which there should not be, at least at the highest levels, any hierarchical structure and then a symmetry of both moving and opening capabilities should be expected.

In this paper we have only considered an essential kernel language for representing mobility, leaving out in particular the operators for restriction and communication. Their introduction should be possible following the lines of [10].

Ambients and Processes

Let $\mathcal{A}$ be a set of *ambient names* ranged over by $a, b, c \ldots$ and $\mathcal{M}$ be the set of *actions*, ranged over by $\mathsf{m}, \mathsf{n}, \ldots$, containing $in\, a,\ out\, a,\ open\, a$ and $so\, a$ for all ambients $a \in \mathcal{A}$. The set $\mathcal{P}$ of *processes* (ranged over by $P,\ Q,\ R, \ldots$) is defined by

$$\mathcal{P} \ ::= \ \mathbf{0} \ \mid \ \mathcal{M}.\mathcal{P} \ \mid \ \mathcal{A}[\mathcal{P}] \ \mid \ \mathcal{P} \,\big|\, \mathcal{P} \ \mid \ !\mathcal{P}.$$

We assume that "." takes precedence over "|". So $\mathsf{m}.\alpha \,\big|\, \beta$ is read $(\mathsf{m}.\alpha)|\beta$. We will write m as short for $\mathsf{m}.\mathbf{0}$.

As customary, the relation of structural congruence $\equiv$ is defined as the minimal reflexive, transitive and symmetric relation which is a congruence and moreover:

- satisfies $!P \equiv !P \,\big|\, P$;
- makes the operator $\big|$ commutative, associative, with $\mathbf{0}$ as zero element.

The behavior of processes is represented by the reduction relation defined in Figure 1. Note that the so action allows a process to open its enclosing ambient. This operation is orthogonal to the other ones ($in\ , out\ , open$) and cannot be internally simulated in the standard Ambient Calculus.

Remark 2.1 Cardelli and Gordon in [7] discuss the primitive *acid* whose reduction rule is:

$$a[acid.P \,\big|\, Q] \rightarrow P \,\big|\, Q$$

They reject *acid* since it allows to entrap an ambient into a location it can never exit. For instance the process $(\nu k)[k[\,] \,\big|\, a[in\, b.acid.in\, k]]$ entraps the ambient b since:

$$(\nu k)(k[\mathbf{0}] \,\big|\, a[in\, b.acid.in\, k]) \,\big|\, b[P] \rightarrow^* (\nu k)k[b[P]]$$

Now, owing to the restriction (νk), $b[P]$ can have no more interactions with the rest of the environment and, in particular, cannot move from k. Similarly we can use so to entrap an ambient b:

$$(\nu k)(k[\mathbf{0}] \,\big|\, a[in\, b.so\, a.in\, k]) \,\big|\, b[P] \rightarrow^* (\nu k)k[b[P]]$$

$$
\begin{array}{lllll}
(red\ in) & a[in\, b.P \mid Q] \mid b[R] & \rightarrow & b[a[P \mid Q] \mid R] \\
(red\ out) & a[b[out\, a.P \mid Q] \mid R] & \rightarrow & a[R] \mid b[P \mid Q] \\
(red\ selfopen) & a[so\, a.P \mid Q] & \rightarrow & P \mid Q \\
(red\ open) & open\, a.P \mid a[Q] & \rightarrow & P \mid Q \\
(R-par) & P \rightarrow Q \quad \Rightarrow & P \mid R \rightarrow Q \mid R & \\
(R-amb) & P \rightarrow Q \quad \Rightarrow & a[P] \rightarrow a[Q] & \\
(R-\equiv) & P' \equiv P' \quad P \rightarrow Q \quad Q \equiv Q' \quad \Rightarrow & P' \rightarrow Q' &
\end{array}
$$

Fig. 1. Reduction

the only difference being that so requires as argument the name a of the ambient to be open from inside. We think that entrapping can be avoided introducing a suitable type system which forbids occurrences of ambients which are arguments of so inside the scope of restrictions.

Observational Equivalence

In the ambient calculus the natural candidates to represent observables are the ambients. The following definition of observational preorder takes the notion of observable proposed in the original system [15].

Definition 2.2

(i) We say that process P *exhibits* an ambient a, notation $P \Downarrow a$ if $P \rightarrow^* a[Q] \mid R$ for some processes $Q,\ R$.

(ii) $P \sqsubseteq Q$ if for all context $\mathcal{C}[\ ]$ and ambients a: $\mathcal{C}[P] \Downarrow a \ \Rightarrow\ \mathcal{C}[Q] \Downarrow a$.

(iii) $P \cong Q$ if $P \sqsubseteq Q$ and $Q \sqsubseteq P$.

Remark 2.3 Note that $P \rightarrow Q$ implies $Q \sqsubseteq P$, but in general $P \not\cong Q$. For instance let $P_1 = open\, a \mid a[b[\mathbf{0}]]$ and $P_2 = b[\mathbf{0}]$. Then $P_1 \rightarrow P_2$ but $P_1 \not\cong P_2$ (take $\mathcal{C}[\ ]$ as $[-]$).

3 Types

Like in type assignment systems for polymorphic λ-calculus, types are seen as properties of type free objects rather than domains in which objects live. Types are intended to provide partial informations about the processes they are associated with. Our language of types must be expressive enough to completely characterize process behaviors. We need so to consider both the ambient, action and parallel

composition as type constructors. Moreover the intersection type constructor is added to represent nondeterminism.

The set $\mathcal{T}$ of *types* (ranged over by $\alpha,\ \beta,\ \gamma, \ldots$) is then defined by

$$\mathcal{T} \ ::= \ \omega \ \mid \ \mathcal{M}.\mathcal{T} \ \mid \ \mathcal{A}[\mathcal{T}] \ \mid \ \mathcal{T} \,\big|\, \mathcal{T} \ \mid \ \mathcal{T} \wedge \mathcal{T}.$$

Type ω represents a property that is true of all processes. Note that $\alpha \,\big|\, \beta$ is the type of a process which can show both properties α and β in two components running in parallel, while $\alpha \wedge \beta$ is the type of a process which can show, in a nondeterministic way, both properties α and β but, in general, along different reduction paths. We assume that $\wedge$ has the lowest precedence.

In connecting types to processes we must consider two distinct formal systems. One is to represent the logical structure of types, determined by their entailment relation (denoted $\leq$), and one to assign types to processes.

The logical structure of types is formalized as a partial order relation representing entailment. We write $\alpha \leq \beta$ to mean that property α entails property β. We write $\alpha \simeq \beta$ if $\alpha \leq \beta \leq \alpha$. Then $\simeq$ is the equivalence relation induced by $\leq$. The formal rules for type entailment are represented in Figure 2.

Note that, as pointed out in Remark 2.3, the execution of an action corresponds to a loss of capabilities. This is formalized by the axioms of the group "Reduction". Rule $(out-in)$ takes into account the fact that, in rule $(red\, in)$, after the consumption of the $in\, a$ action, the process inside a is always allowed to perform a sequence $out\, a,\ in\, a$ of actions. A similar motivation holds for rule $(in - out)$. Intersection represent "may" nondeterminism. A process has type $\alpha \wedge \beta$ if it can possibly exhibit both property α and property β. Axiom $(. \,\big|_2)$ is crucial to represent this. Axioms $(\omega 1)$, $(\omega 2)$ and rule $(cg - \,\big|\,)$ imply that $\alpha \,\big|\, \beta \leq \alpha$, i.e. parallel composition corresponds to increase of capabilities. Moreover using $(\wedge - id)$, $(\omega 1)$, $(\,\big|\, 1)$, $(cg - \,\big|\,)$, $(\wedge - \leq)$, $(\omega 2)$, we get

$$\alpha \,\big|\, \beta \leq \alpha \,\big|\, \beta \wedge \alpha \,\big|\, \beta \leq \alpha \,\big|\, \omega \wedge \omega \,\big|\, \beta \leq \alpha \wedge \beta,$$

i.e. capabilities in parallel are "better" than capabilities in alternative.

Types will be always considered modulo $\simeq$. Note that $\simeq$ is preserved by both intersection and parallel composition with ω. The operators $\,\big|\,$ and $\wedge$ are associative so, for instance, we can write unambiguously $\alpha \,\big|\, \beta \,\big|\, \gamma$. Parallel composition of types are also considered modulo permutations, and intersection of types are considered modulo permutations and repetitions (rules $(\wedge - id)$, $(\wedge - l)$, $(\wedge - r)$).

A parallel composition $\alpha_1 \,\big|\, \ldots \,\big|\, \alpha_n$ will sometimes be denoted by $\overrightarrow{\alpha}$ in *vector* notation. An intersection of types $\alpha_1 \wedge \ldots \wedge \alpha_n$ will be denoted by $\bigwedge_{i\in\{1\ldots n\}} \alpha_i$. In this case $\beta \propto \bigwedge_{i\in\{1\ldots n\}} \alpha_i$ denotes that $\beta \equiv \alpha_i$ for some $i \in \{1 \cdots n\}$.

A crucial technical notion is that of normal type.

Definition 3.1 (i) The set $\mathcal{N} \subset \mathcal{T}$ of *normal* types is defined inductively in the following way:

(a) $\omega \in \mathcal{N}$.

- Commutativity and distributivity of $\mid$

$$(\mid 1) \quad \alpha \mid \beta \simeq \beta \mid \alpha \qquad\qquad (\mid 2) \quad (\alpha \mid \beta) \mid \gamma \simeq \alpha \mid (\beta \mid \gamma)$$

- Axioms for ω

$$(\omega 1) \quad \alpha \leq \omega \qquad\qquad (\omega 2) \quad \alpha \simeq \alpha \mid \omega$$

- Distributivity of $\wedge$

$$([\,]\wedge) \quad a[\alpha \wedge \beta] \simeq a[\alpha] \wedge a[\beta] \qquad\qquad (\mid \wedge) \quad \alpha \mid (\beta \wedge \gamma) \simeq (\alpha \mid \beta) \wedge (\alpha \mid \gamma)$$

$$(.\wedge) \quad \mathsf{m}.(\alpha \wedge \beta) \simeq \mathsf{m}.\alpha \wedge \mathsf{m}.\beta$$

- Sequentialization

$$(. \mid_1) \quad \mathsf{m}.\alpha \mid \beta \leq \mathsf{m}.(\alpha \mid \beta) \qquad (. \mid_2) \quad \mathsf{m}.\alpha \mid \mathsf{n}.\beta \simeq \mathsf{m}.(\alpha \mid \mathsf{n}.\beta) \wedge \mathsf{n}.(\mathsf{m}.\alpha \mid \beta)$$

- Reduction

$$(in) \quad a[in\, b.\alpha \mid \beta] \mid b[\gamma] \leq b[a[\alpha \mid \beta] \mid \gamma] \qquad (out) \quad a[b[out\, a.\alpha \mid \beta] \mid \gamma] \leq a[\gamma] \mid b[\alpha \mid \beta]$$

$$(selfopen) \quad a[so\, a.\, \alpha \mid \beta] \leq \alpha \mid \beta \qquad (open) \quad open\, a.\alpha \mid a[\beta] \leq \alpha \mid \beta$$

$$(out{-}in) \quad in\, a.out\, a.in\, a.\alpha \leq in\, a.\alpha \qquad (in{-}out) \quad out\, a.in\, a.out\, a.\alpha \leq out\, a.\alpha$$

- Congruence

$$(cg-[\,]) \ \frac{\alpha \leq \beta}{a[\alpha] \leq a[\beta]} \qquad (cg-act) \ \frac{\alpha \leq \beta}{\mathsf{m}.\alpha \leq \mathsf{m}.\beta} \qquad (cg-\mid) \ \frac{\alpha \leq \gamma \quad \beta \leq \delta}{\alpha \mid \beta \leq \gamma \mid \delta}$$

- Transitivity

$$(trans) \quad \frac{\alpha \leq \beta \quad \beta \leq \gamma}{\alpha \leq \gamma}$$

- Logical

$$(\wedge - id) \quad \alpha \leq \alpha \wedge \alpha \qquad\qquad (\wedge - l) \quad \alpha \wedge \beta \leq \alpha$$

$$(\wedge - r) \quad \alpha \wedge \beta \leq \beta \qquad\qquad (\wedge - \leq) \quad \frac{\alpha \leq \alpha' \quad \beta \leq \beta'}{\alpha \wedge \beta \leq \alpha' \wedge \beta'}$$

Fig. 2. Type Entailment Rules

(b) $\omega \mid \phi \in \mathcal{N}$ where $\phi \in \mathcal{N}$.
(c) $\mathsf{m}.\phi \in \mathcal{N}$ where $\phi \in \mathcal{N}$.
(d) $a[\phi] \in \mathcal{N}$ where $\phi \in \mathcal{N}$.
(e) $\phi \mid a[\psi] \in \mathcal{N}$ where $\phi, \psi \in \mathcal{N}$.

(ii) A normal type is *easy* if is either ω or of the form (c).

Also normal types are seen modulo permutations, repetitions and parallel composition with ω. Let $\phi, \psi, \xi, \chi...$ range over normal types. In general a normal type different from ω has the form $\phi \mid a_1[\psi_1] \mid \ldots \mid a_n[\psi_n]$ (or $\phi \mid \overrightarrow{a[\psi]}$ in vector notation) where ϕ is easy or is missing (or, equivalently, is ω) and $n \geq 0$.

Normal types do not contain intersections. A normal type represents a process in which, in each ambient a, there is at most one action that can be possibly performed. Nondeterminism is left, however, since different actions can be enabled at the same time in different ambients.

Definition 3.2 Let $\simeq_0$ be the equivalence relation defined by the rules obtained by replacing $\leq$ by $\simeq_0$ in the rules $(\mid 1)$, $(\mid 2)$, $(\omega 2)$, $([\,]\wedge)$, $(\mid \wedge)$, $(.\wedge)$, $(. \mid_2)$ $(trans)$ of Figure 2.

We can show by structural induction on types that each type is $\simeq_0$ equivalent, modulo permutation and parallel composition with ω, to a unique type which is an intersection of normal types. This equivalence can be obtained by repeatedly "pushing out" intersections, essentially by means of $([\,]\wedge)$, $(\mid \wedge)$a and $(.\wedge)$.

Lemma 3.3 *For all $\alpha \in \mathcal{T}$ there are normal types $\phi_1, \ldots, \phi_n$ ($n \geq 1$) such that $\alpha \simeq_0 \bigwedge_{i\in\{1\ldots n\}} \phi_i$. The type $\bigwedge_{i\in\{1\ldots n\}} \phi_i$ is called the* normal form *of α, denoted $nf(\alpha)$, and it is unique modulo permutations and parallel compositions with ω.*

Ambients are inactive with respect to normal forms in the following sense.

Lemma 3.4 *Let $\phi, \overrightarrow{a[\psi]}$ be normal types. Then $\phi \propto nf(\alpha)$ iff $\phi \mid \overrightarrow{a[\psi]} \propto nf(\alpha \mid \overrightarrow{a[\psi]})$.*

Lemma 3.5 *Let $nf(\alpha) = \bigwedge_{i\in I} \phi_i$. Then*

(i) $nf(a[\alpha]) = \bigwedge_{i\in I} a[\phi_i]$.
(ii) $nf(\mathsf{m}.\alpha) = \bigwedge_{i\in I} \mathsf{m}.\phi_i$.
(iii) *Let $nf(\beta) = \bigwedge_{j\in J} \psi_j$. Then $nf(\alpha \mid \beta) = \bigwedge_{i\in I, j\in J} nf(\phi_i \mid \psi_j)$.*
(iv) $nf(nf(\alpha) \mid \beta) = nf(\alpha \mid \beta)$.

The entailment relation can be specialized to normal types. Let $\leq_N \subset \mathcal{N} \times \mathcal{N}$ denote this relation, defined by the rules of Figure 3.

In the rules for $\leq_N$ the r.h.s. is naturally a normal type whenever the l.h.s. is normal, except for rules $(open^N)$ and $(selfopen^N)$, since the parallel composition of two normal types is not normal, in general. Note that in rule $(selfopen^N)$ the normal type ψ can be ω or, equivalently, it can be missing.

We need some technical definitions and lemmas.

- Commutativity and distributivity of $\mid$

$$(\mid 1^N) \quad \phi \mid \psi \simeq_N \psi \mid \phi \quad \text{provided } \phi \mid \psi \text{ is normal}$$

$$(\mid 2^N) \quad (\phi \mid \psi) \mid \xi \simeq_N \phi \mid (\psi \mid \xi) \quad \text{provided } (\phi \mid \psi) \mid \xi \text{ is normal}$$

- Axioms for ω

$$(\omega 1^N) \quad \phi \leq_N \omega \qquad\qquad (\omega 2^N) \quad \phi \mid \omega \simeq_N \phi$$

- Sequentialization

$$(. \mid_1^N) \quad \mathsf{m}.\phi \mid \overrightarrow{a[\psi]} \leq \mathsf{m}.(\phi \mid \overrightarrow{a[\psi]})$$

- Reduction

$$(in^N) \qquad a[in\, b.\phi \mid \overrightarrow{c[\psi]}] \mid b[\xi] \leq_N b[a[\phi \mid \overrightarrow{c[\psi]}] \mid \xi]$$

$$(out^N) \qquad a[b[out\, a.\phi \mid \overrightarrow{c[\psi]}] \mid \xi] \leq_N a[\xi] \mid b[\phi \mid \overrightarrow{c[\psi]}]$$

$$(selfopen^N) \qquad a[so\, a.\phi \mid \overrightarrow{c[\chi]}] \mid \psi \leq_N \xi \mid \overrightarrow{c[\chi]} \quad \text{for all } \xi \propto nf(\phi \mid \psi)$$

$$(open^N) \qquad open\, a.\phi \mid a[\psi] \leq_N \xi \quad \text{for all } \xi \propto nf(\phi \mid \psi)$$

$$(out{-}in^N) \qquad in\, a.out\, a.in\, a.\phi \leq in\, a.\phi$$

$$(in{-}out^N) \qquad out\, a.in\, a.out\, a.\phi \leq out\, a.\phi$$

- Congruence

$$(cg{-}[\,]^N) \quad \frac{\phi \leq_N \psi}{a[\phi] \leq_N a[\psi]} \qquad\qquad (cg{-}action^N) \quad \frac{\phi \leq_N \psi}{\mathsf{m}.\phi \leq_N \mathsf{m}.\psi}$$

$$(cg{-}\mid^N) \quad \frac{\phi \leq_N \phi' \quad \psi \leq_N \psi'}{\phi \mid a[\psi] \leq_N \phi' \mid a[\psi']}$$

- Transitivity

$$(trans^N) \quad \frac{\phi \leq_N \psi \quad \psi \leq_N \xi}{\phi \leq_N \xi}$$

Fig. 3. Entailment Rules for Normal Types

Definition 3.6(i) A *characteristic pair* (c.p. for short) is a pair $\langle \mathsf{m}\,,\, \overrightarrow{a[\phi]}\rangle$ where m is an action and $\overrightarrow{a[\phi]}$ is a parallel composition of ambients containing normal types.

(ii) A *characteristic sequence* (c.s. for short) is a sequence $\mathcal{S} = \mathcal{P}_1, \ldots, \mathcal{P}_n$ of c.p..

Definition 3.7 Let ϕ be an easy type. The c.s. *associated* to ϕ, $\mathcal{S}^\phi$ is defined in the following way:

(i) $\mathcal{S}^\omega = \epsilon$

(ii) $\mathcal{S}^{\mathsf{m}.(\phi \,|\, \overrightarrow{a[\psi]})} = \langle \mathsf{m}\,,\, \overrightarrow{a[\psi]}\rangle \cdot \mathcal{S}^\phi$ where ϕ is easy and $\cdot$ denotes concatenation (if $\overrightarrow{a[\psi]}$ has no elements we take it as ω).

Note that if $\phi = \mathsf{m}_1.(\mathsf{m}_2.(\mu \,|\, \overrightarrow{b[\xi]}) \,|\, \overrightarrow{a[\psi]})$ then $\mathcal{S}^\phi = \langle \mathsf{m}_1,\, \overrightarrow{a[\psi]}\rangle \cdot \langle \mathsf{m}_2,\, \overrightarrow{b[\xi]}\rangle \cdot \mathcal{S}^\mu$ and so on. It is easy to see that there is a one-one correspondence between easy types and c.s.. If $\mathcal{S}$ is a c.s. let $\tau^{\mathcal{S}}$ denote the easy type corresponding to $\mathcal{S}$. So $\tau^{\mathcal{S}^\phi} \equiv \phi$ and vice-versa.

Let $\mathcal{S}_1,\, \mathcal{S}_2$ be c.s., the *shuffle* product $\mathcal{S}_1||\mathcal{S}_2$ is the set of all their possible interleavings. The proof of the following property is routine.

Lemma 3.8 *Let $\phi,\, \psi$ be easy types. Then $\xi \propto nf(\phi \,|\, \psi)$ iff $\mathcal{S}^\xi \in \mathcal{S}^\phi||\mathcal{S}^\psi$.*

A crucial lemma representing the entailment properties of normal types is the following.

Lemma 3.9 *Let $\phi,\, \psi,\, \xi,\, \chi$ be normal types such that $\phi \leq_N \psi$ and $\xi \leq_N \chi$. Then for all $\nu \propto nf(\psi \,|\, \chi)$ there is $\mu \propto nf(\phi \,|\, \xi)$ such that $\mu \leq_N \nu$.*

Proof. It is enough to prove the lemma assuming $\phi \leq_N \psi$ and $\xi \equiv \chi$. The general property can be easily obtained by transitivity. The proof is then by induction on the proof of $\phi \leq_N \psi$. The most difficult cases are when $\leq_N$ has been obtained by rules $(open^N)$ and $(selfopen^N)$. We discuss the former case. The proof of the latter is similar.

Let $\phi = open\, a.\rho \,|\, a[\sigma]$ and $\psi \propto nf(\rho \,|\, \sigma)$. By rule $(open^N)$ we have $open\, a.\rho \,|\, a[\sigma] \leq_N \psi$. Using Lemma 3.4 it can be assumed w.l.o.g. that $\rho,\, \sigma$ and ξ are easy, and in this case also ψ must be easy. Moreover, by Lemma 3.4 we assume that $\xi = \mathsf{m}.\xi'$, otherwise the proof is trivial. Let now

$$\zeta \propto nf(\psi \,|\, \xi)$$

i.e., by Lemma 3.5(4), $\zeta \propto nf(\rho \,|\, \sigma \,|\, \xi)$. This means that $\mathcal{S}^\zeta \in \mathcal{S}^\rho||\mathcal{S}^\sigma||\mathcal{S}^\xi$. Moreover let $\mathcal{S}_1$ be the c.s. obtained by eliminating from $\mathcal{S}^\zeta$ the elements of $\mathcal{S}^\sigma$, in the same order in which they occur in $\mathcal{S}^\sigma$. Then we must have $\tau^{\mathcal{S}_1} \propto nf(\rho \,|\, \xi)$, and $\zeta \propto nf(\sigma \,|\, \tau^{\mathcal{S}_1})$.

Now let's note that, by Lemma 3.5(2), for each type $\mu \propto nf(\rho \,|\, \xi)$ we have that $open\, a.\mu \,|\, a[\sigma] \propto nf(open\, a.\rho \,|\, a[\sigma] \,|\, \xi)$.
In fact, by $(.\,|_2)$, $open\, a.\rho \,|\, a[\sigma] \,|\, \xi \simeq_0 open\, a.(\rho \,|\, \xi) \,|\, a[\sigma] \;\wedge\; \alpha$ for some type α.

$$(\omega) \quad \vdash P : \omega \qquad (\mathsf{m}) \quad \frac{\vdash P : \alpha}{\vdash \mathsf{m}.P : \mathsf{m}.\alpha}$$

$$(amb) \quad \frac{\vdash P : \alpha}{\vdash a[P] : a[\alpha]} \qquad (\,|\,) \quad \frac{\vdash P : \alpha \quad \vdash Q : \beta}{\vdash P \,|\, Q : \alpha \,|\, \beta}$$

$$(!) \quad \frac{\vdash P : \alpha \quad \vdash !P : \beta}{\vdash !P : \alpha \,|\, \beta} \qquad (\leq) \quad \frac{\vdash P : \alpha \quad \alpha \leq \beta}{\vdash P : \beta}$$

Fig. 4. Type Inference Rules

So $open\, a.\tau^{\mathcal{S}_1} \,|\, a[\sigma] \propto nf(open\, a.\rho \,|\, a[\sigma] \,|\, \xi)$, and then

$$open\, a.\tau^{\mathcal{S}_1} \,|\, a[\sigma] \leq_N \zeta$$

since $\zeta \propto nf(\sigma \,|\, \tau^{\mathcal{S}_1})$. □

The main lemma of this section relates normal forms and $\leq_N$ to $\leq$.

Lemma 3.10 *Let $\alpha \leq \beta$. Then for all $\psi \propto nf(\beta)$ there exists $\phi \propto nf(\alpha)$ such that $\phi \leq_N \psi$.*

Proof. By induction on the proof of $\alpha \leq \beta$. The most difficult case is that of rule $(cg - \,|\,)$ which is handled using Lemmas 3.9 and 3.5(3). □

Corollary 3.11 *Let ϕ, ψ be normal types. Then $\phi \leq \psi$ iff $\phi \leq_N \psi$.*

4 Type Inference

Being our types strongly similar to processes it is to very natural to devise type assignment rules. They are represented in Figure 4. Let $\vdash^-$ denote inference in the system obtained from the rules of Figure 4 by eliminating $(\leq)$.

Notice that the system $\vdash$ only has introduction rules for the various constructors. Elimination rules are replaced by rule $(\leq)$, which also introduces and eliminates $\wedge$ (see Proposition 4.5).

The following is a sort of normal form for deductions: the proof is by induction on deductions.

Lemma 4.1 *If $\vdash P : \alpha$ then there is a type α' such that $\vdash^- P : \alpha'$ and $\alpha' \leq \alpha$.*

As usual we can prove by a simple induction on deductions a generation lemma.

Lemma 4.2 (Generation Lemma) (i) $\vdash \mathbf{0} : \alpha$ *iff* $\alpha \simeq \omega$;

(ii) $\vdash m.P : \alpha$ *iff* $\vdash^- P : \beta$ *and* $m.\beta \leq \alpha$ *for some* β*;*

(iii) $\vdash a[P] : \alpha$ *iff* $\vdash^- P : \beta$ *and* $a[\beta] \leq \alpha$ *for some* β*;*

(iv) $\vdash P \mid Q : \alpha$ *iff* $\vdash^- P : \beta, \vdash^- Q : \gamma$ *and* $\beta \mid \gamma \leq \alpha$ *for some* β, γ*;*

(v) $\vdash !P : \alpha$ *iff* $\vdash^- P : \beta_i$ $(1 \leq i \leq n)$ *and* $\beta_1 \mid \ldots \mid \beta_n \leq \alpha$ *for some* $\beta_1, \ldots, \beta_n$.

Previous lemma says that the types of a term can be obtained in a uniform way from the types of its subterms, and this will guarantee the compositionality of the filter model we will build in the next section.

Since we are in a "may" perspective, it is natural that a process P offer all the ambients offered by one of its reducts Q (may be more). At the type assignment level this means that types are preserved under subject expansion. Of course subject reduction should not hold; for example, the reduction of a process P with the rule $(red\ open)$ produce a process that in general offers less ambients (and so has less types). Instead congruent processes have the same types. Both properties can be proved by induction on the definitions of $\equiv$ and $\rightarrow^*$ using Lemma 4.2.

Lemma 4.3 (Subject Congruence) $P \equiv Q$ *and* $\vdash Q : \alpha \quad \Rightarrow \quad \vdash P : \alpha.$

Lemma 4.4 (Subject Expansion) $P \rightarrow^* Q$ *and* $\vdash Q : \alpha \quad \Rightarrow \quad \vdash P : \alpha.$

Lastly we can shown by structural induction on processes using Lemma 4.2 that the standard rule of intersection introduction

$$(\wedge I) \quad \frac{\vdash P : \alpha \quad \vdash P : \beta}{\vdash P : \alpha \wedge \beta}$$

is admissible in our system.

Proposition 4.5 (Admissibility of $(\wedge I)$**)** $\vdash P : \alpha$ *and* $\vdash P : \beta$ *imply* $\vdash P : \alpha \wedge \beta$*, i.e. rule* $(\wedge I)$ *is admissible.*

5 The Filter Model

We capitalize on the type assignment system of previous section for defining a filter model of the ambient calculus. We mainly follow the development line of [10].

Let $\langle D; \sqsubseteq \rangle$ be a preorder. A subset L of D is a *filter* if L is a non-empty upper set, i.e., $l \in L$ and $l \sqsubseteq l'$ imply $l' \in L$, and every finite subset of L has a greatest lower bound in L.

Consider the set $\mathcal{T}$ of types with the inclusion $\leq$ defined in Section 3. The greatest lower bound of a finite non-empty set of types is the intersection of the types in the set.

We can observe that the set $\mathcal{F}(\mathcal{T})$ of filters over $\mathcal{T}$ is a model of $\mathcal{P}$ in the following sense. For all P define

$$[\![P]\!] = \{\alpha \in \mathcal{T} \mid \vdash P : \alpha\}.$$

From rules (ω), $(\leq)$ and the admissibility of $(\wedge)$ we have that $[\![P]\!] \in \mathcal{F}(\mathcal{T})$ for all P. Subject expansion can now be rephrased into the following statement:

$$\text{if } P \rightarrow^* Q \text{ then } [\![P]\!] \supseteq [\![Q]\!].$$

The filter model is naturally ordered by subset inclusion. The inclusion on filters induces an ordering on terms.

Definition 5.1 Let $P, Q \in \mathcal{P}$. $P \sqsubseteq_F Q$ if and only if $[\![P]\!] \subseteq [\![Q]\!]$.

The order relation $\sqsubseteq_F$ can be easily characterized by means of the deducibility of types as follows.

Proposition 5.2 *Let $P, Q \in \mathcal{P}$. $P \sqsubseteq_F Q$ if and only if, for all α, $\vdash P : \alpha$ implies $\vdash Q : \alpha$.*

We will prove that the filter model exactly mirrors the operational semantics, i.e., that it is adequate and complete, i.e. it is fully abstract.

Adequacy

The adequacy proof requires a double induction on types and deductions. Following a standard methodology, we split this induction by introducing a realizability interpretation of types as sets of terms. The underlying idea is that a process P belongs to the interpretation of a type α if and only if α can be derived for P.

First we give an interpretation of normal types, and then we build the interpretation of all types, taking into account Lemmas 3.3 and 3.10. In defining the interpretation of types we will use a somewhat stronger notion of reduction over processes.

Definition 5.3 The reduction relation $\rightsquigarrow$ over $\mathcal{P}$ is defined by adding to the rules of Fig. 1 the following rules:

$$\begin{array}{ll}
(seq) & \mathsf{m}.P \,|\, Q \rightsquigarrow \mathsf{m}.(P \,|\, Q) \\
(red-out{-}in) & in\, a.out\, a.in\, a.P \rightsquigarrow in\, a.P \\
(red-in{-}out) & out\, a.in\, a.out\, a.P \rightsquigarrow out\, a.P \\
(R-act) & P \rightsquigarrow^* Q \;\Rightarrow\; \mathsf{m}.P \rightsquigarrow^* \mathsf{m}.Q
\end{array}$$

It is easy to verify that $\rightsquigarrow$ does not modify the notion of convergence, i.e. that $P \Downarrow a$ iff if $P \rightsquigarrow^* a[Q] \,|\, R$ for some processes Q, R.

A standard induction on the definition of $\rightsquigarrow$ shows that the subject expansion property holds also with respect to $\rightsquigarrow$ reductions.

Lemma 5.4 $P \rightsquigarrow^* Q$ *and* $\vdash Q : \alpha \;\Rightarrow\; \vdash P : \alpha$.

The interpretation of normal types as sets of terms is given by structural induction.

Definition 5.5 The interpretation of normal types is defined by:

(i) $[\![\omega]\!] = \mathcal{P}$

(ii) $[\![\mathsf{m}\,.\phi]\!] = \{P \mid P \rightsquigarrow^* \mathsf{m}\,.Q \text{ such that } Q \in [\![\phi]\!]\}$.

(iii) $[\![a[\phi]]\!] = \{P \mid P \rightsquigarrow^* a[Q] \,\big|\, R \text{ such that } Q \in [\![\phi]\!]\}$.

(iv) $[\![\phi \,\big|\, a[\psi]]\!] = \{P \mid P \rightsquigarrow^* Q \,\big|\, a[R] \text{ such that } Q \in [\![\phi]\!] \text{ and } R \in [\![\psi]\!]\}$.

We need to prove the soundness of the normal type inclusion relation with respect to the interpretation of normal types. To this aim we need the following Lemma, whose proof is similar to that of Lemma 3.9.

Lemma 5.6 *Let ϕ, ψ be normal types. Then $P \in [\![\phi]\!]$ and $Q \in [\![\psi]\!]$ imply $P \,\big|\, Q \in \xi$ for all $\xi \propto nf(\phi \,\big|\, \psi)$.*

The soundness of the type inclusion relation can be shown by induction on $\leq_N$ definition. The most interesting case are axioms $(open^N)$ and $(selfopen^N)$, which can be handled using Lemma 5.6.

Lemma 5.7 *Let ϕ, ψ be normal types. Then $\phi \leq_N \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

We can now define the interpretation of all types.

Definition 5.8 The interpretation of arbitrary types is defined by:

$$[\![\alpha]\!] = \bigwedge_{\phi \propto nf(\alpha)} [\![\phi]\!].$$

From Lemmas 3.10 and 5.7 we get the soundness of the type inclusion relation with respect to the interpretation of types.

Lemma 5.9 *If $\alpha \leq \beta$ then $[\![\alpha]\!] \subseteq [\![\beta]\!]$.*

As expected the type interpretation perfectly matches the type assignment system.

Theorem 5.10 (Soundness and completeness of $\vdash$) *$\vdash P : \alpha$ iff $P \in [\![\alpha]\!]$.*

Proof. Soundness is proved by induction on the derivation of $\vdash P : \alpha$, using Lemma 5.9 for rule $(\leq)$.
As for completeness, by definition it sufficies to show that if $P \in [\![\phi]\!]$ then $\vdash P : \phi$, when ϕ is normal. This proof can be done by structural induction on ϕ using Subject Expansion with respect to $\rightsquigarrow^*$ (Lemma 5.4). □

Now we are able to characterize convergency by means of typing.

Lemma 5.11 (Resource property) *$\vdash P : a[\omega]$ iff $P \Downarrow a$.*

Proof. $\vdash P : a[\omega]$ iff (by Theorem 5.10) $P \in [\![a[\omega]]\!]$ iff (by Definition 5.5) $P \rightsquigarrow a[Q] \,|\, R$ for some processes Q, R. □

We can now conclude the adequacy proof.

Theorem 5.12 (Adequacy) *If $P \sqsubseteq_F Q$ then $P \sqsubseteq Q$.*

Proof. If $\mathcal{C}[P] \Downarrow a$ then by Lemma 5.11 we get $\vdash \mathcal{C}[P] : a[\omega]$. This together with $P \sqsubseteq_F Q$ imply $\vdash \mathcal{C}[Q] : a[\omega]$, so we can conclude $\mathcal{C}[Q] \Downarrow a$ using again Lemma 5.11. □

Completeness

Our completeness proof relies on building processes $T_\phi^{x,y}$, where ϕ is a normal type and x, y are fresh ambient names with respect to ϕ. Their intended behavior is such that, for all normal types ϕ, $x[P] \,|\, T_\phi^{x,y} \Downarrow y$ iff $\vdash P : \phi$. The process P under testing is formerly enclosed in an ambient x for technical convenience.

In building these terms it is useful to have a process which converges to z iff it is in parallel with a process which converges to both x and y.

Lemma 5.13 *Let w be a fresh ambient name. Let's define*

$$H^{x,y \Rightarrow z} = w[in\, x.out\, x.in\, y.out\, y.z[out\, w]].$$

Then $H^{x,y\Rightarrow z} \,|\, P \Downarrow z$ iff $P \Downarrow x$ and $P \Downarrow y$.

The processes T are defined by structural induction on normal types.

In defining them we assume to have an unlimited source of ambient names, and to be able to pick new ambients names without clashing with the ambient names occurring in the processes we are testing.

Definition 5.14 [Test Terms] Let ϕ be a normal type and x, y ambient names.The processes $T_\phi^{x,y}$ are defined inductively on ϕ in the following way:

- $T_\omega^{x,y} = p[in\, x.out\, x.y[out\, p]]$
- $T_{in\, a.\phi}^{x,y} = a[p[in\, x.so\, p.out\, a.in\, v.in\, z]] \,|\, v[z[open\, x.t[out\, z.out\, v.open\, v.open\, a]]] \,|\, open\, t \,|\, T_\phi^{z,y}$
- $T_{out\, a.\phi}^{x,y} = p[in\, x.so\, p.in\, v.in\, a.in\, z] \,|\, v[a[z[open\, x.t[out\, z.out\, v.open\, v.open\, a]]]] \,|\, open\, t \,|\, T_\phi^{z,y}$
- $T_{open\, a.\phi}^{x,y} = p[in\, x.so\, p.a[in\, v.in\, z]] \,|\, v[z[open\, x.t[out\, z.out\, v.open\, v]]] \,|\, open\, t \,|\, T_\phi^{z,y}$
- $T_{so\, a.\phi}^{x,y} = p[in\, x.so\, p.in\, v.in\, z.in\, a] \,|\, v[z[a[open\, x.t[out\, z.out\, v.open\, v]]]] \,|\, open\, t \,|\, T_\phi^{z,y}$

- $T^{x,y}_{\phi \,|\, a[\psi]} = p[in\, x.in\, a.so\, p.out\, x.in\, v.in\, w] \,|\, v[w[open\, a.t[out\, w.out\, v.open\, v]]] \,|\, open\, t \,|\, T^{x,q}_{\phi} \,|\, T^{w,z}_{\psi} \,|\, H^{q,z \Rightarrow y}$
- $T^{x,y}_{a[\psi]} = T^{x,y}_{\omega \,|\, a[\psi]}$
- $T^{x,y}_{\omega \,|\, \phi} = T^{x,y}_{\phi}$

where we assume that all ambient names ($p, q, v, w, x, y, z, \ldots$, except a) introduced in the definition of each $T^{x,y}_{\phi}$ are fresh. We call them the *extra* names of $T^{x,y}_{\phi}$, denoted $EN(T^{x,y}_{\phi})$.

Note that all the terms $T^{x,y}_{\phi}$ are reducible only if they are running in parallel with an ambient which exhibit x at top level and that their reduction is needed to produce a process which exhibits y at top level. So all of them must interact with x in the proper way to do the job, as the following Lemma shows.

Lemma 5.15 *Let ϕ be a normal type and Q be a process containing no occurrences of any ambient name belonging to $EN(T^{x,y}_{\phi})$ different from x. Then*

$$Q \,|\, T^{x,y}_{\phi} \Downarrow y \quad \textit{implies} \quad Q \rightarrow^* x[P] \,|\, Q' \textit{ and } \vdash P : \phi.$$

Proof. The proof is by induction on the normal type ϕ. If $\phi \equiv \omega$ the proof is easy. The induction step is by cases on ϕ. As a sample we give the case in which ϕ is $in\, a.\phi$. The proof of the other cases is similar.

Let

$$S = a[p[in\, x.so\, p.out\, a.in\, v.in\, z]] \,|\, v[z[open\, x.t[out\, z.out\, v.open\, v.open\, a]]] \,|\, open\, t$$

then $T^{x,y}_{in\, a.\phi} = S \,|\, T^{z,y}_{\phi}$. Since $Q \,|\, S \,|\, T^{z,y}_{\phi} \Downarrow y$, by induction hypothesis we have

$$(1) \qquad Q \,|\, S \rightarrow^* z[P] \,|\, Q'$$

where $\vdash P : \phi$. By construction all names occurring in S cannot occur in Q except x and a. Then note that to have (1) the following facts are needed.

- v must be opened, there is no way for z to exit v. But v can be opened only if $open\, v$ is exercised.
- This is possible only by allowing ambient t to exit at top level, exercising the actions $out\, z.out\, v$, and being opened. But this is possible only if $open\, x$ is exercised first.
- To exercise $open\, x$ we must have ambient x in z, but only p, after entering x and selfopening, can move x inside v and z.
- Now p must run in parallel to x to enter it. This is possible only if Q exhibits at top level ambient x containing an $in\, a$ action, i.e.

$$(2) \qquad Q \rightarrow^* x[in\, a.Q_1 \,|\, Q_2] \,|\, Q_3.$$

After p enters x and selfopen, x can exit a and enter v and z. Note that if a is instead opened by Q by an $open\, a$ action we can also have x and p running in parallel, but in this case Q, which does not know p, has no possibility of putting again p into an ambient named a to allow the $out\, a$ action to be exercised.

- No other interaction between reductions in Q and reductions in S are possible.

By (2) and the previous points then we must have:

$$Q \,|\, S \rightarrow^* x[in\, a.Q_1 \,|\, Q_2] \,|\, Q_3 \,|\, S \rightarrow^* z[P] \,|\, Q'$$

where $Q_1 \,|\, Q_2 \rightarrow^* P$. By the subject expansion Lemma 4.4 and the generation Lemma 4.2 we then have $\vdash Q_1 : \alpha_1$ and $\vdash Q_2 : \alpha_2$ where $\alpha_1 \,|\, \alpha_2 \leq \phi$. This implies $\vdash in\, a.Q_1 : in\, a.\alpha_1$ and, by $in\, a.\alpha_1 \,|\, \alpha_2 \leq in\, a.(\alpha_1 \,|\, \alpha_2)$ (axiom $(. \,|\, _1)$), we get $\vdash in\, a.Q_1 \,|\, Q_2 : in\, a.\phi$. □

As an immediate corollary we get then the following Lemma.

Lemma 5.16 *Let P a process not containing occurrences of names belonging to $EN(T_\phi^{x,y})$. Then $x[P] \,|\, T_\phi^{x,y} \Downarrow y$ implies $\vdash P : \phi$.*

Using the soundness and completeness of $\vdash$ (Theorem 5.10) and Lemma 5.16 one can finally show that the given processes characterize typing.

Theorem 5.17 (Characterization) *Let P a process not containing occurrences of names belonging to $EN(T_\phi^{x,y})$. Then $T_\phi^{x,y} \,|\, x[P] \Downarrow y$ iff $\vdash P : \phi$.*

Completeness now follows easily.

Theorem 5.18 (Completeness) *If $P \sqsubseteq Q$ then $P \sqsubseteq_F Q$.*

Proof. If $P \not\sqsubseteq_F Q$ then there is a type α such that $\vdash P : \alpha$ and $\not\vdash Q : \alpha$. Then by Lemmas 3.3, 3.10, and rule $(\leq)$ there is a normal type ϕ such that $\vdash P : \phi$ and $\not\vdash Q : \phi$. By Theorem 5.17 we get that $T_\phi^{x,y} \,|\, x[P] \Downarrow y$ and $T_\phi^{x,y} \,|\, x[Q] \not\Downarrow y$. So we conclude $P \not\sqsubseteq Q$. □

6 Final Remarks

We have defined a filter model $\mathcal{F}(\mathcal{T})$ for the Selfopening Ambient Calculus which is fully abstract with respect to the notion of contextual equivalence defined in [15]. Being our language a proper extension of the standard Ambient Calculus we have immediately that $\mathcal{F}(\mathcal{T})$ is also a model of the standard Ambient Calculus. This model is adequate ($P \sqsubseteq_F Q$ implies $P \sqsubseteq Q$) but not fully abstract as the following counterexample shows. Take

$$P_1 = a[b[out\, c]]$$

$$P_2 = a[b[open\, d]] \,|\, d[in\, b.out\, c]$$

The process P_1 and P_2 are uncomparable in the model (they have different types) but operationally, in the standard Ambient Calculus, we have $P_1 \sqsubseteq P_2$. In fact both exhibit the ambient a which contains only an ambient b. To show that $P_1 \not\sqsubseteq P_2$ we should find a context allowing to exercise the action $out\, c$ in P_1. But to obtain this we need to enclose b in an ambient c and this is possible only if we eventually open a. In this case, in a may perspective, we cannot avoid that d jumps into b and is opened there, allowing P_2 to show the same behavior as P_1. Only using the so action we are able to build a context that separates P_1 and P_2.

We are aware that so can cause undesired behaviours, but we are confident that a suitable type discipline can avoid them. The design of such a discipline will be subject of further investigations.

References

[1] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51(1-2):1–77, 1991.

[2] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[3] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940, 1983.

[4] G. Boudol. Lambda-calculi for (strict) parallel functions. *Information and Computation*, 108(1):51–127, 1994.

[5] M. Bugliesi and G. Castagna. Secure safe ambients. In *POPL'01*, pages 222–235. ACM Press, 2001.

[6] L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for mobile ambients. In *ICALP'99*, volume 1644 of *LNCS*, pages 230–239, Berlin, 1999. Springer-Verlag.

[7] L. Cardelli and A. D. Gordon. Mobile ambients. In *FoSSaCS'98*, volume 1378 of *LNCS*, pages 140–155, Berlin, 1998. Springer-Verlag.

[8] L. Cardelli and A. D. Gordon. Types for mobile ambients. In *POPL'99*, pages 79–92, New York, 1999. ACM Press.

[9] L. Cardelli and A. D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *POPL'00*, pages 365–377. ACM Press, 2000.

[10] F. Damiani, M. Dezani-Ciancaglini, and P. Giannini. A filter model for mobile processes. *Mathematical Structures in Computer Science*, 9(1):63–101, 1999.

[11] P. Degano, F. Levi, and C. Bodei. Safe ambients: Control flow analysis and security. In *ASIAN'00*, volume 1961 of *LNCS*, pages 199–214, Berlin, 2000. Springer-Verlag.

[12] M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. Finite models for conjunctive-disjunctive λ-calculi. *Theoretical Computer Science*, 170(1–2):83–128, 1996.

[13] M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. A filter model for concurrent λ-calculus. *SIAM Journal on Computing*, 27(5):1376–1419, 1998.

[14] M. Dezani-Ciancaglini and I. Salvo. Security types for safe mobile ambients. In *ASIAN'00*, volume 1961 of *LNCS*, pages 215–236, Berlin, 2000. Springer-Verlag.

[15] A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. In *FoSSaCS'99*, volume 1578 of *LNCS*, pages 212–226, Berlin, 1999. Springer-Verlag.

[16] R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In *SAS'99*, volume 1694 of *LNCS*, pages 134–148, Berlin, 1999. Springer-Verlag.

[17] C. Hartonas and M. Hennessy. Full abstractness for a functional/concurrent language with higher-order value-passing. *Information and Computation*, 145(1):64–106, 1998.

[18] M. Hennessy. A fully abstract denotational model for higher-order processes. *Information and Computation*, 112(1):55–95, 1994.

[19] M. Hennessy. Higher-order process and their models. In *ICALP'94*, volume 820 of *LNCS*, pages 286–303, Berlin, 1994. Springer-Verlag.

[20] C.-H. L. Ong. Non-determinism in a functional setting. In *LICS'93*, pages 275–286, Montreal, Canada, 1993. IEEE Computer Society Press.

[21] D. S. Scott. Domains for denotational semantics. In *ICALP'82*, volume 140 of *LNCS*, pages 577–613, Berlin, 1982. Springer-Verlag.